Fourier Transform and Its Application

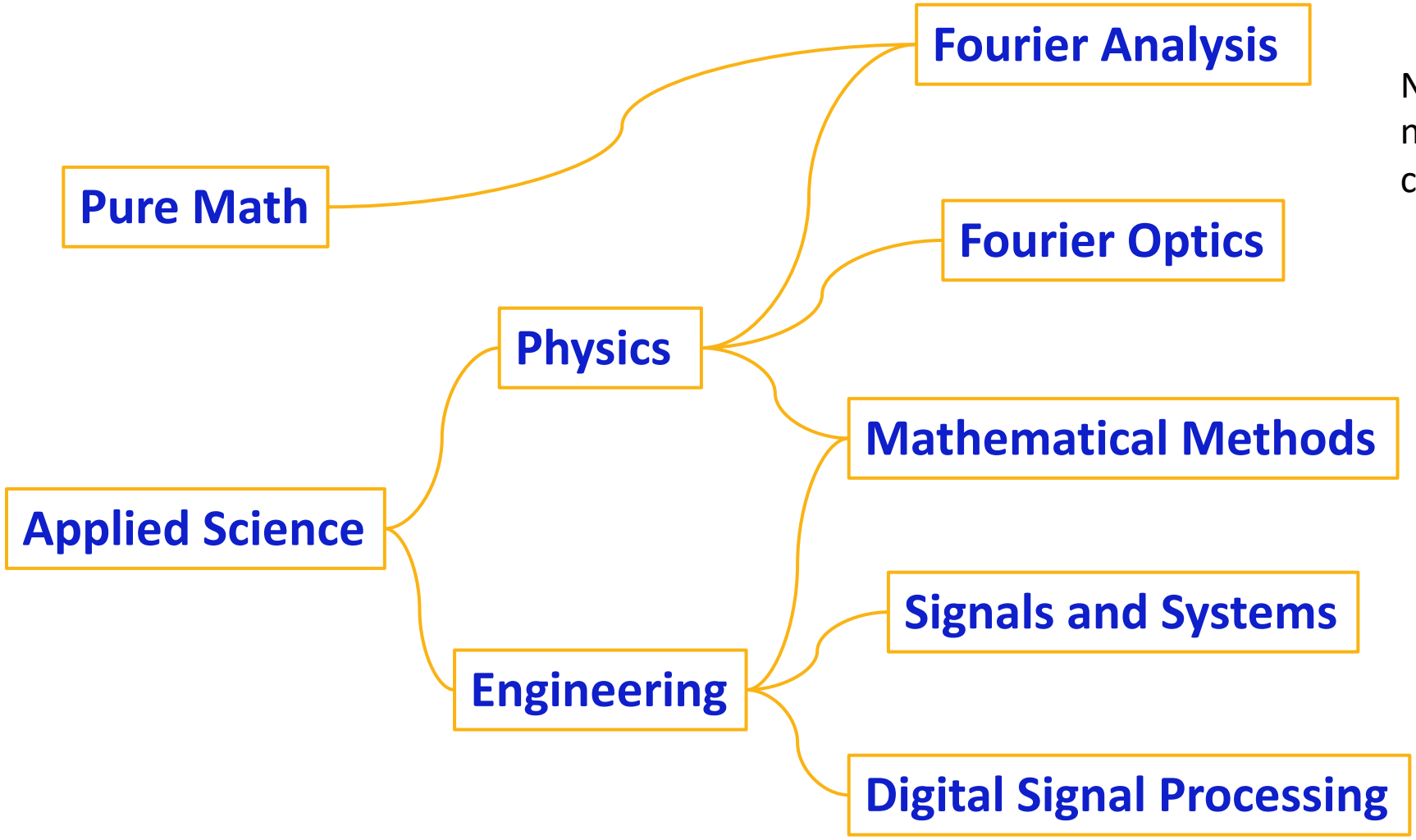
Chapter 0: Hands-On Introduction

Jake W. Liu

Outline

- **Introduction**
  - Different approaches to learning FT
- **FFT in Action**
  - Implementing FFT in Julia
  - Visualizing time-domain and frequency-domain signals
- **Understanding FFT Output**
  - Complex values and magnitude interpretation
  - Identifying signal components from FFT results
- **Improving Frequency Analysis**
  - Effect of sampling rate and signal length
  - Resolving frequency components accurately
- **Practical Considerations**
  - Nyquist limit and aliasing
  - Scaling and normalization in FFT
- **Summary**

Different Approaches to Learn FT



Neither accurate nor exhaustive characterization 😜

Goal of This Introduction

- Learn FT from a programming point of view (practical)
- Prove basic properties from known examples
- Hands-on introductions (learn to use FFT first)
  - Find Fourier coefficients of periodic signals
  - Approximate Fourier transforms

An FFT Example

- We are using Julia as the programming language. Two packages are used (for plotting and FFT)

```
using PlotlySupply
using FFTW
```

- Suppose we have a signal with the form

```
s(t) = 0.5 * sin(2pi * t / 24) + sin(2pi * t / 24 / 7) + 1.5 # whatever vs. hour
```

- Function **s(t)** can represent any physical phenomena. Let it be “my daily energy”.

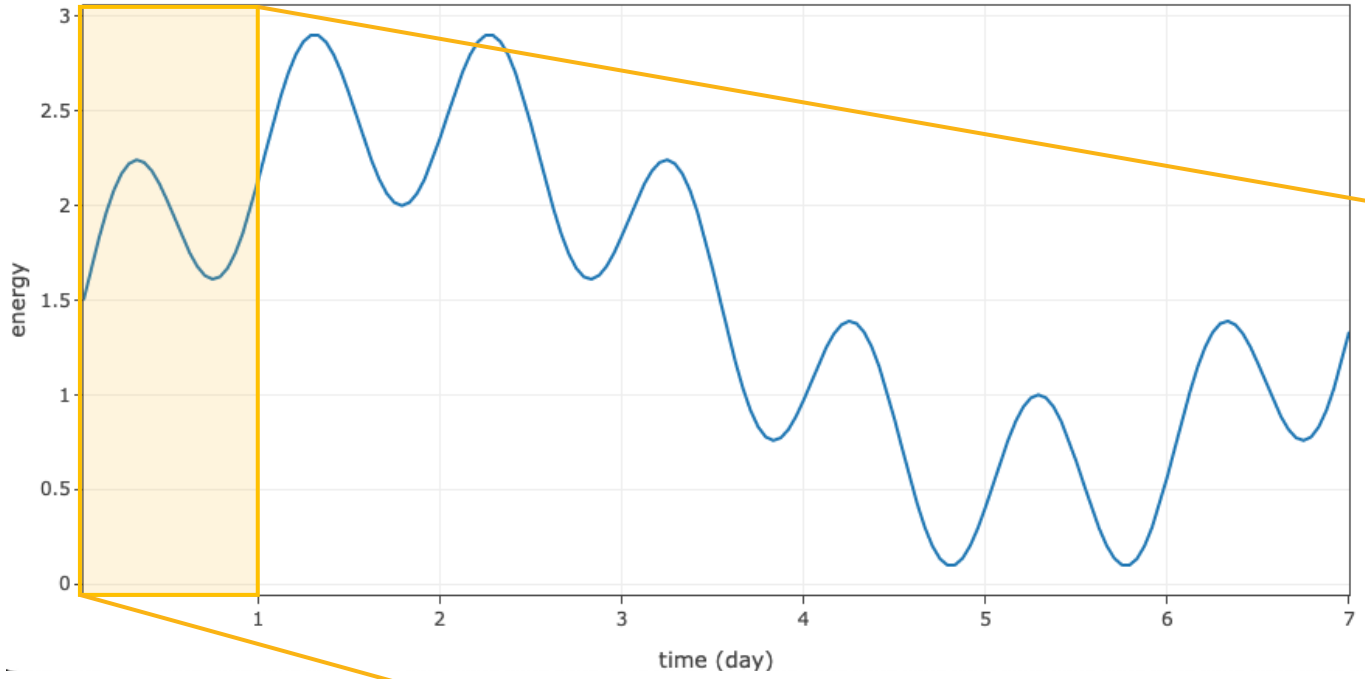
An FFT Example

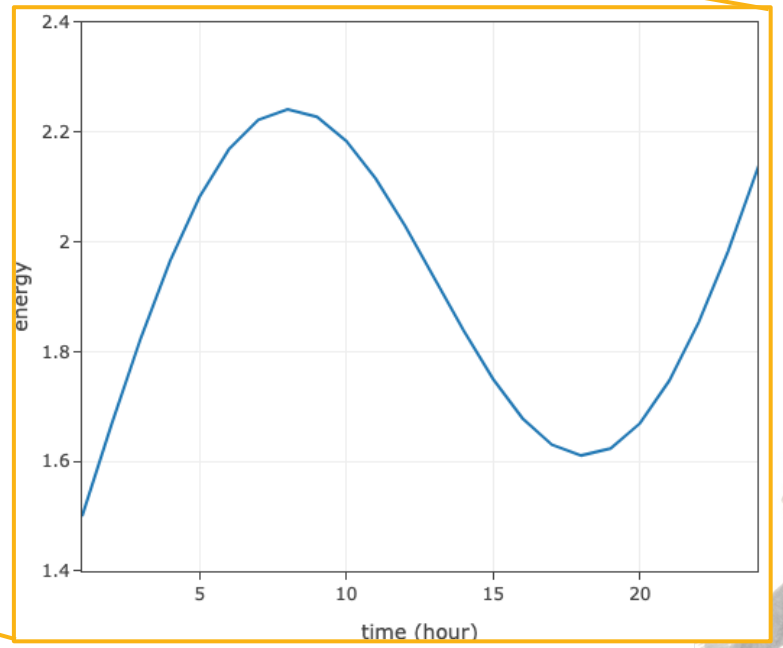
- Let’s plot the signal in a weekly scale and daily scale:

```
hour = collect(1:24*7*1)
day = hour ./ 24
eng = s.(t)

ylabel = "energy"
fig1 = plot_scatter(day, eng; xlabel = "time (day)", ylabel = ylabel)
display(fig1)

fig2 = plot_scatter(hour, eng;
  xlabel = "time (hour)",
  ylabel = ylabel,
  xrange = [1, 24],
  yrange = [1.4, 2.4])
display(fig2)
```

An FFT Example

$$s(t) = 0.5\sin(2\pi t / 24) + \sin(2\pi t / 168) + 1.5$$

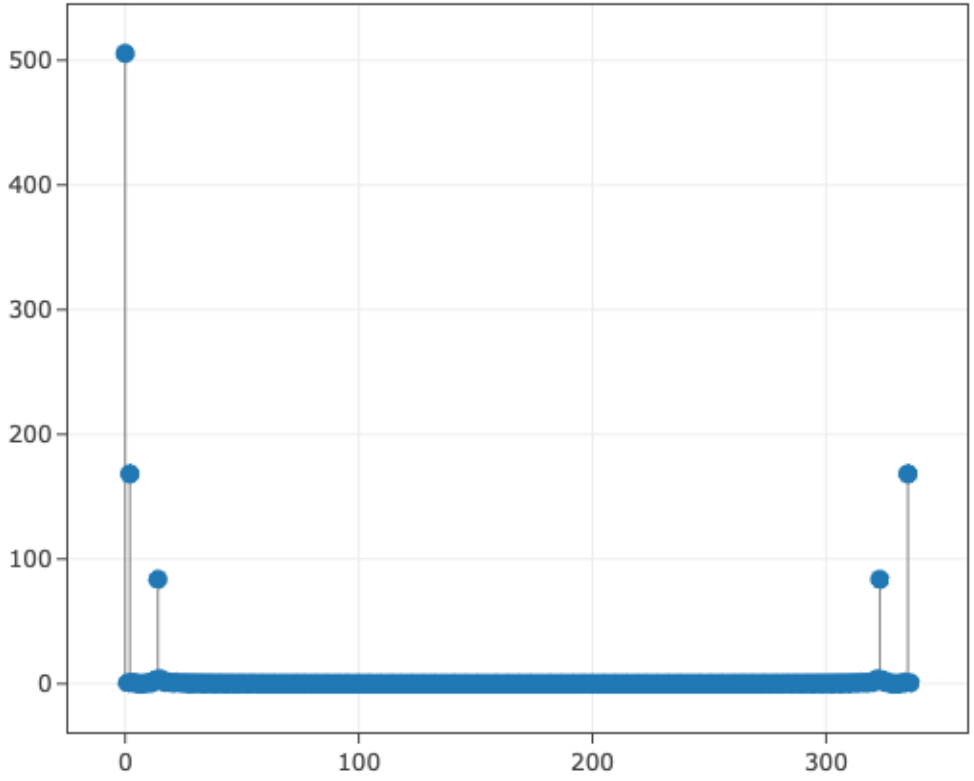




An FFT Example

- Suppose you know nothing about FFT… Before understanding any mathematics, let's plot a FFTed result first

```
engf = fft(eng)

fig3 = plot_stem(abs.(engf)) # what is this???
display(fig3)
```



- Some observations:
  - **engf** is complex valued

    (thus the absolute value is plotted)
  - We still don't know the meaning of x-axis
  - FFT result (**engf**) ⇔ **s(t)**'s coefficient?

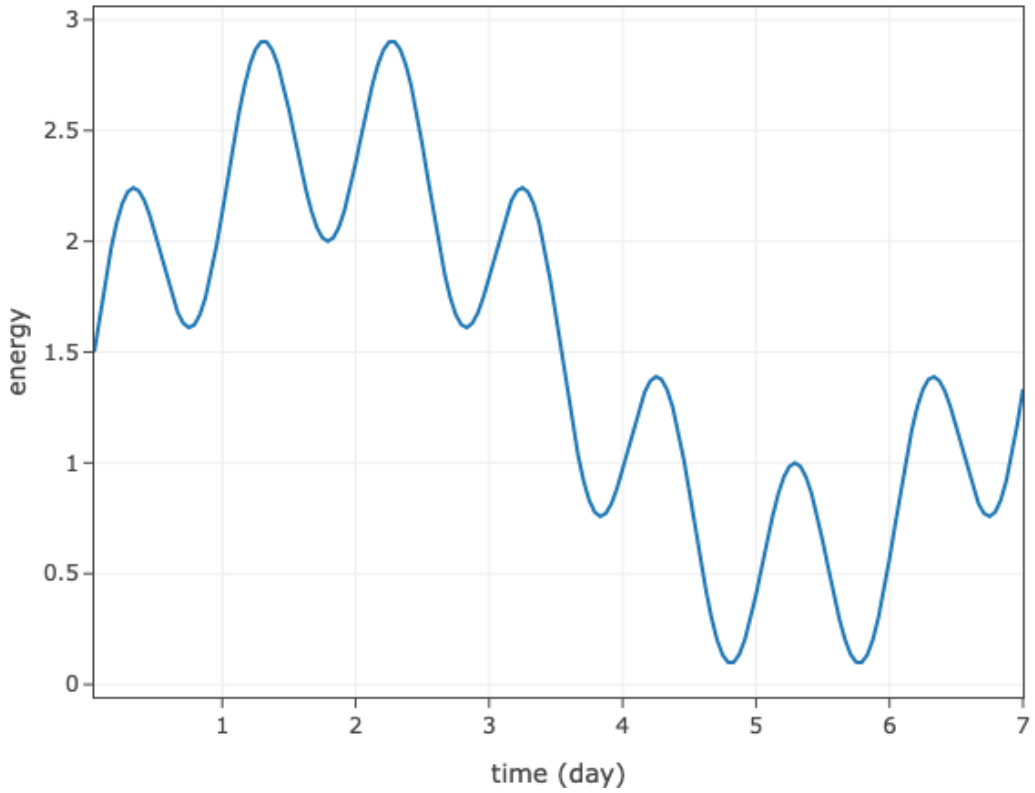
An FFT Example

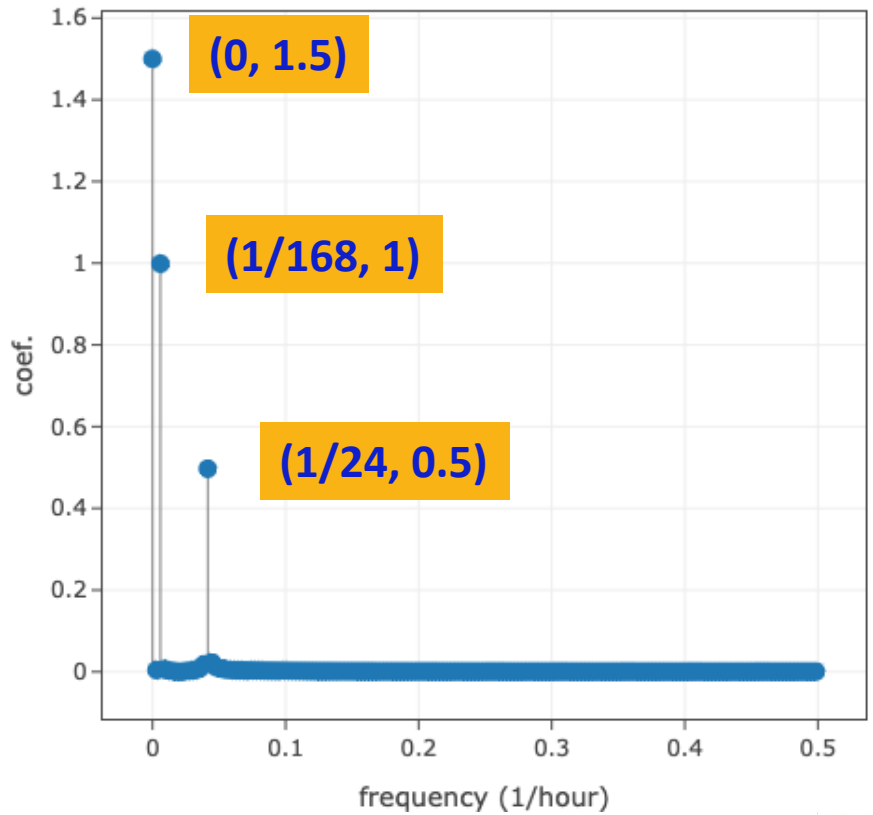
- Let's do some magic... Our goal is the sort out the coefficients of **s(t)** from the FFT result **engf**. Maybe after some try and error 😂:

```
# our goal is to find the coefficient 0.5 & 1 in s(t) from engf
n = length(engf)
engf ./= n
srate = 1 ./ (hour[2] - hour[1])
if isodd(n)
  fac = (n - 1) / n
else
  fac = 1
end
f = LinRange(0, srate / 2 * fac, floor(Int64, n / 2 + 1))
fsig = engf[1:length(f)]
fsig[2:end] .= 2 .* abs.(fsig[2:end])

fig4 = plot_stem(f, abs.(fsig), xlabel = "frequency (1/hour)", ylabel = "coef.")
display(fig4)
```

An FFT Example

$$s(t) = 0.5\sin(2\pi t / 24) + \sin(2\pi t / 168) + 1.5$$





Single-sided Fourier spectrum

Finding Fourier Coefficients with FFT

- Okay, let's try to write a function that returns the single-sided Fourier spectrum and the frequency span (given the signal and sampling rate)

```
function fft_fs(sig, srate = 1)
  Sig = fft(sig)
  N = length(Sig)
  Sig ./= N
  if isodd(N)
    f = LinRange(0, srate / 2 * (N - 1) / N, floor(Int64, N / 2 + 1))
  else
    f = LinRange(0, srate / 2 * 1, floor(Int64, N / 2 + 1))
  end

  sSig = Sig[1:length(f)]
  sSig[2:end] .= 2 .* abs.(sSig[2:end])

  return sSig, f
end
```

Finding Fourier Coefficients with FFT

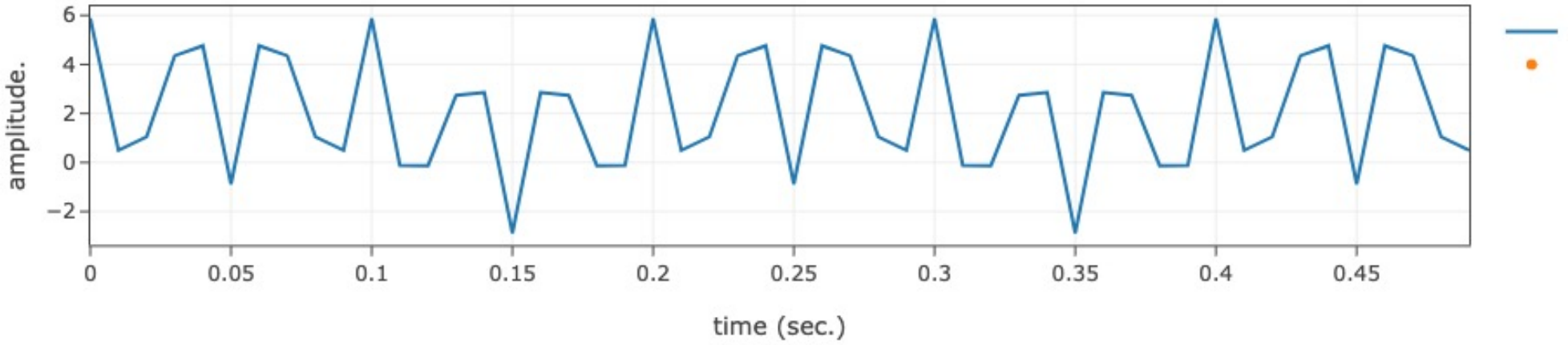
- A simple sinusoidal example:

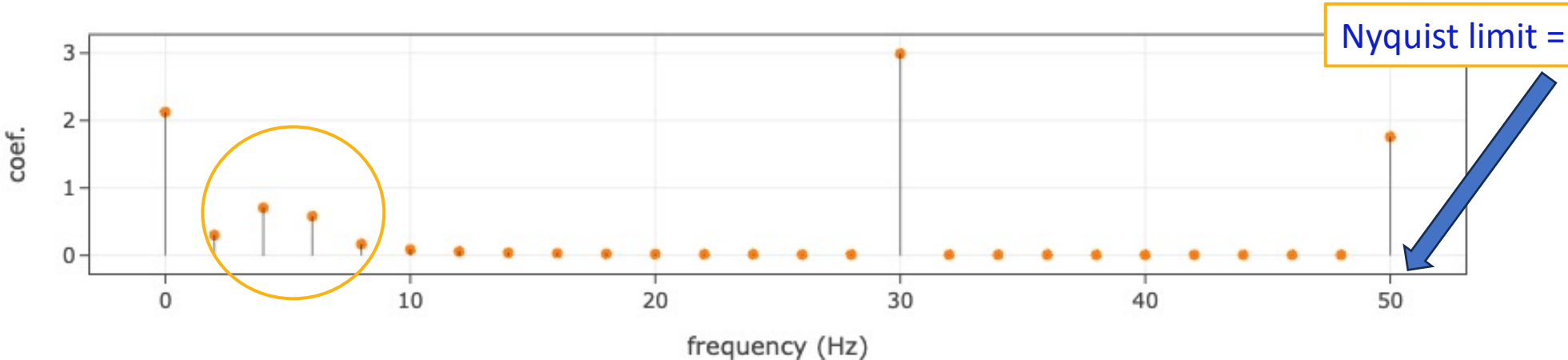
```
srate = 100
time = collect(0 : 1/srate : 0.5 - 1/srate)
N = length(time)

# boring sinusoids
s1(t) = 2 + sin(2pi*5*t) + 3 * cos(2pi*30*t) + 1.5 * sin(2pi*50*t + pi/5)
S1, freq = fft_fs(s1.(time), srate)
fig_td = plot_scatter(time, s1.(time), xlabel = "time (sec.)", ylabel = "amplitude.")
fig_fd = plot_stem(freq, abs.(S1), xlabel = "frequency (Hz)", ylabel = "coef.")
fig = [fig_td; fig_fd]
set_template!(fig, :plotly_white)
display(fig)
```

Finding Fourier Coefficients with FFT

$$s(t) = 2 + \sin(2\pi * 5 * t) + 3\cos(2\pi * 30t) + 1.5 * sin(2\pi * 50 * t + \pi/5)$$

- We found out that the component at 5 Hz is not well captured…

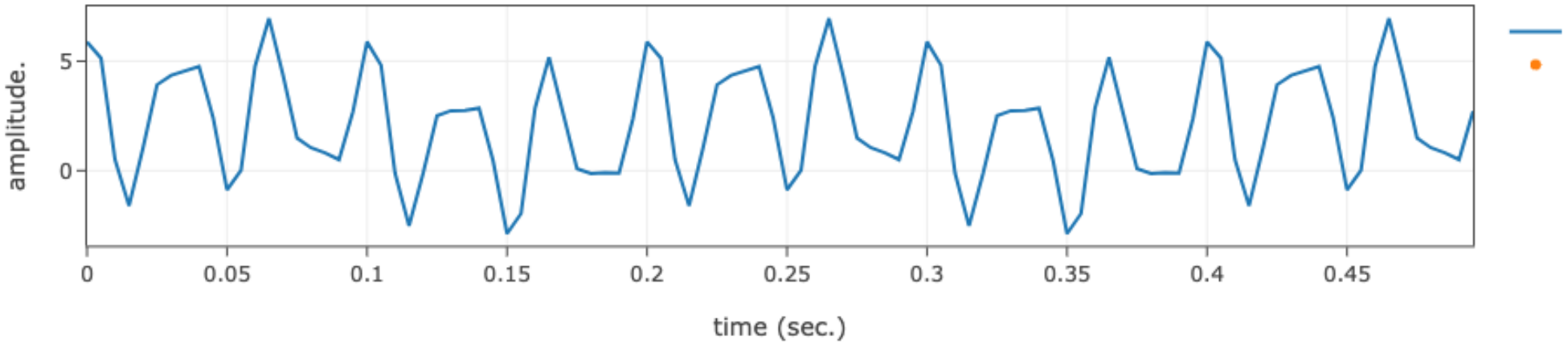


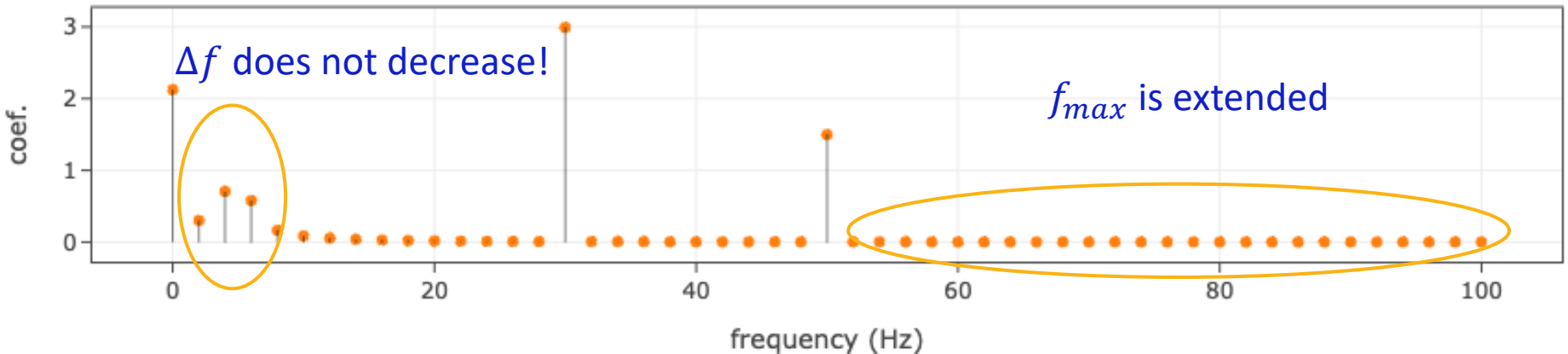


Nyquist limit = sampling rate / 2

Increasing Sampling Rate

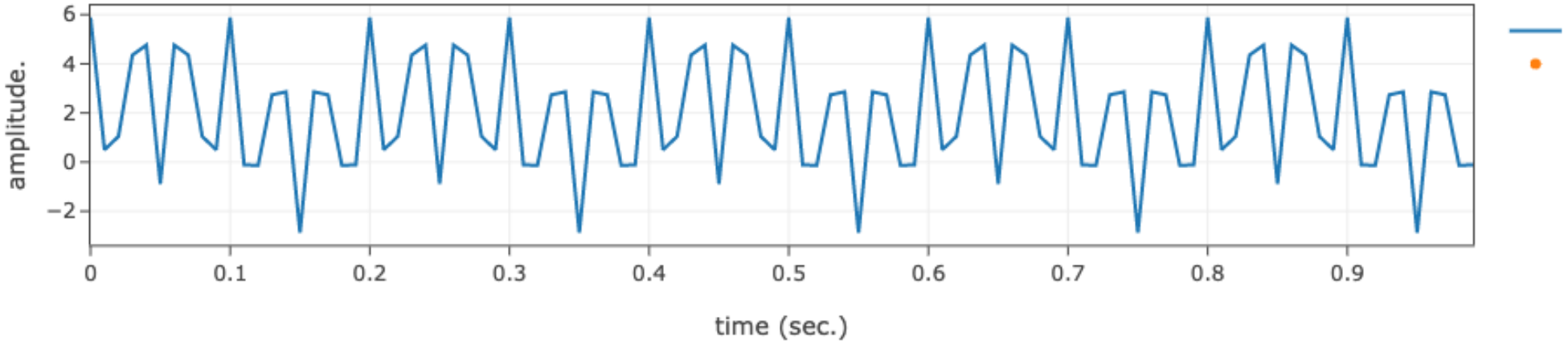
- **Case A**: increase the sampling rate to 200 (decrease Δt)

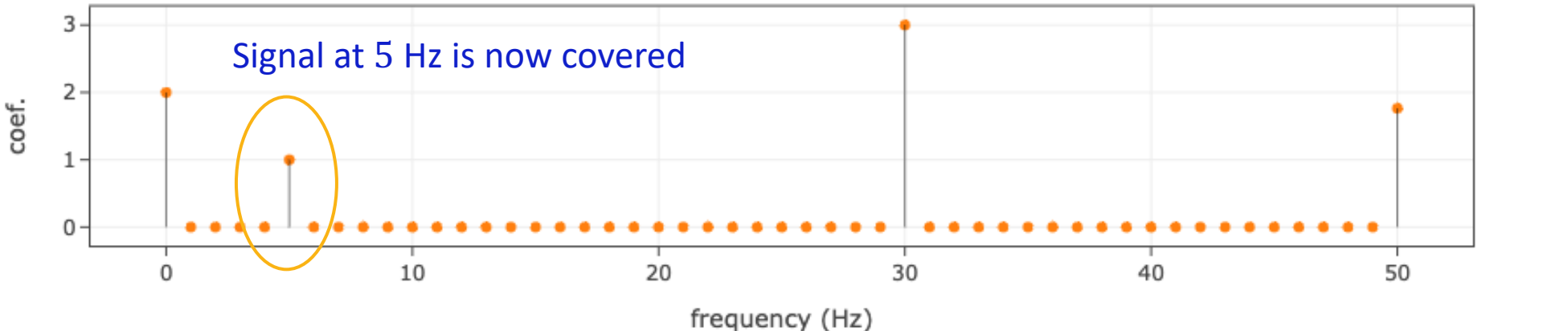




Increasing Signal Length

- **Case B**: extend the total sampling time to 1 (doubles signal length)





Some Observations

- The amplitude of the FFT result is proportional to the signal length N (a normalization is needed)*
- maximum frequency f_{max} equals to sampling rate divided by 2.
- For single-sided spectrum, a factor of 2 (except the DC component) is needed.
- Decreasing Δt increases the f_{max}
- Extending total signal length N decreases Δf

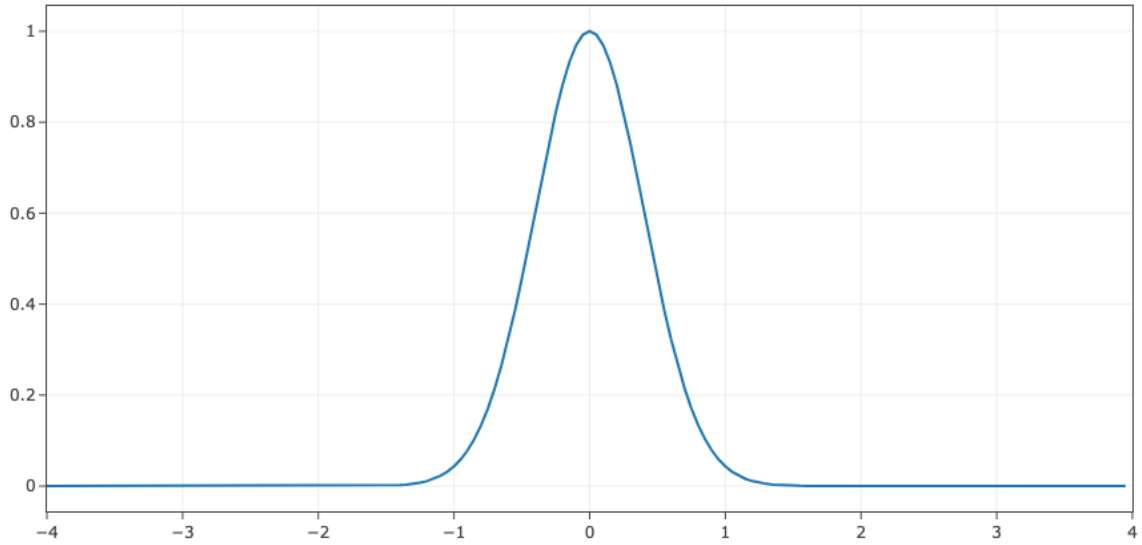
$$\Delta f \Delta t = 1/N$$

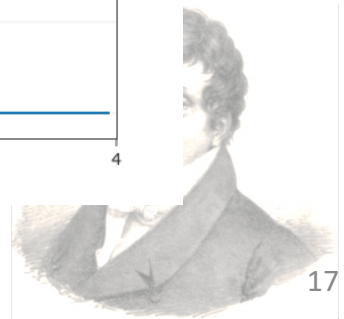
* This actually depends on the definition of DFT.

Approximation of FT with FFT

- We can also approximate Fourier transform with FFT.
- Consider the function $g(t) = e^{-\pi t^2}$, its Fourier transform is known analytically, which has the same form as $g(t)$.

```
srate = 20
dt = 1/srate
tb = 4
t = collect(-tb : 1/srate : tb-1/srate)
sig = exp.(-pi.*t.^2)
N = length(t)

display(plot_scatter(t, sig))
```

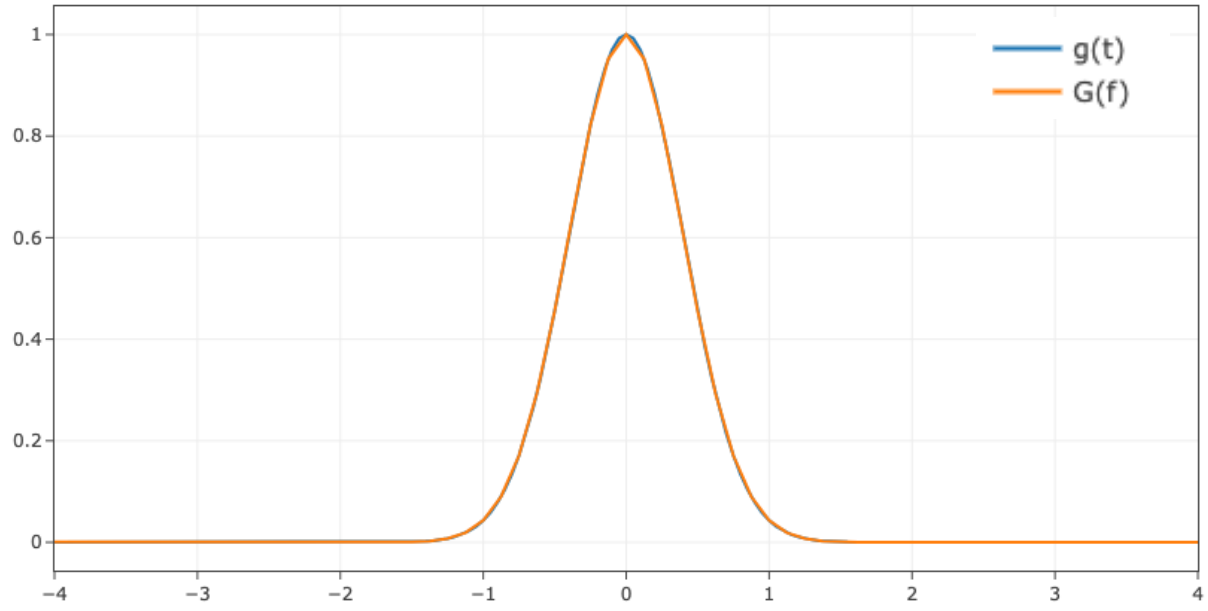




Approximation of FT with FFT

- The proper way to do this is to
  1. **fft** the signal
  2. multiply Δt
  3. conduct **fftshift**

```
f = collect(-N/2:1:N/2-1) .* srate ./ N
sig_dft = zeros(ComplexF64, N)
sig_dft = fft(sig) .* dt
sig_dft .= fftshift(sig_dft)
display(plot_scatter([t,f], [sig, abs.(sig_dft)],
  xrange=[-tb,tb], legend=["g(t)", "G(f)"]))
```



On Frequency Range and Shift

- Frequency range after **fftshift**:

$$\left(-\text{ceil}\left[\frac{N-1}{2}\right]:\text{floor}\left[\frac{N-1}{2}\right]\right)\Delta f$$

- Frequency range before **fftshift**

$$\left(0:\text{floor}\left[\frac{N-1}{2}\right];-\text{ceil}\left[\frac{N-1}{2}\right]:-1\right)\Delta f$$

- Use **fftshift** after **fft** if 0-centered frequency is needed
- Use **ifftshift** before **ifft** if 0-centered frequency is applied

Summary

- Fourier Transform (FT) in Practice
  - Learn by coding first, understand theory later
  - Use FFT to extract frequency components of signals.
- Understanding FFT Results
  - Find Fourier series
  - Approximate Fourier transform
- Improving Frequency Analysis
  - Relationship between sampling rate, resolution and signal length